

# Package ‘ParallABEL’

February 22, 2010

**Title** Parallel of Genome-Wide Association function

**Version** 0.1-0

**Author** Unitsa Sangket

**Description** Support for parallel GenABEL package in R.

**Maintainer** Unitsa Sangket <usangket@yahoo.com>

**License** GPL

**Depends** R (>= 2.8), utils

## R topics documented:

type1.p	1
type2.p	3
type3.p	5
type4.p	7
<b>Index</b>	<b>10</b>

---

type1.p *Parallel for the analyses of statistics of each SNP*

---

## Description

Parallel for the analyses of statistics of each SNP, such as SNP characterization statistics or association test statistics

## Usage

```
type1.p(npro, fun, data, data_f, ...)
```

## Arguments

npro	number of processors on compute nodes
fun	function name will be process such as mlreg.p
data	object of snp.data-class
data_f	file name that was saved the input object of snp.data-class, the object must be named "data"
...	further arguments passed to function of fun argument

**Author(s)**

Unitsa Sangket, Yuri S. Aulchenko and Surakameth Mahasirimongkol

**Examples**

```
Example 1 (submit job on R)
#####
#clear working space
rm(list = ls())
library(GenABEL)
library(ParallABEL)
formula=dm2~sex+age
data(ge03d2.clean)
data <- ge03d2.clean[, ]
npro=2 # npro = number of processors on the compute-nodes
fun=mlreg.p
```

```
output.p = type1.p(npro,fun,data,formula=formula)
```

```
output.p[1:5,]
```

```
mpi.quit(save="no")
```

```
#####
```

```
Example 2 (submit job on Sun Grid Engine)
```

You have to create 2 files if you want to submit a job on Sun Grid Engine.  
(<http://math.acadiau.ca/ACMMaC/Rmpi/submitting.html>)

```
File 1 (R_script.sh):
```

```
#####
#!/bin/bash
```

```
# Run in the current directory
```

```
#$ -cwd
```

```
#$ -j y
```

```
#$ -V
```

```
# Run using bash
```

```
#$ -S /bin/bash
```

```
# The number of processors required - 1 for frontend-node, plus whatever
```

```
# number for compute-nodes.
```

```
# This example runs with a processor on frontend-node, plus 2 processors on compute-nodes
```

```
#$ -pe lam 3
```

```
# Run the job. Replace with whatever R script should run
```

```
lamrun -np 1 R --slave CMD BATCH R_script.R R_script_sh.Rout
```

```
#####
```

```
File 2 (R_script.R):
```

```
#####
```

```
#clear working space
```

```

rm(list = ls())
library(GenABEL)
library(ParallABEL)
formula=dm2~sex+age
data(ge03d2.clean)
data <- ge03d2.clean[, ]
npro=2 # npro = number of processors on the compute-nodes
fun=mlreg.p

output.p = type1.p(npro,fun,data,formula=formula)

output_Rdata ="output_type1_mlreg_c3.Rdata"
save(output.p,file=output_Rdata)
# You can load the output from output_type1_mlreg_c3.Rdata

output.p[1:5,]

mpi.quit(save="no")

#####

For submit the job please use command:
qsub R_script.sh

You can see the progress on R_script_sh.Rout.

```

---

type2.p

*Parallel for the analyses of statistics of each individual*


---

### Description

Parallel for the analyses of statistics of each individual, for example, summary statistics of genotype quality for each sample

### Usage

```
type2.p(npro, fun, data, data_f, ...)
```

### Arguments

npro	number of processors on compute nodes
fun	function name will be processed such as hom
data	object of snp.data-class
data_f	file name that was saved the input object of snp.data-class, the object must be named "data"
...	further arguments passed to function of fun argument

### Author(s)

Unitsa Sangket, Yurii S. Aulchenko and Surakameth Mahasirimongkol

**Examples**

```

Example 1 (summit job on R)
#####
#clear working space
#clear working space
rm(list = ls())
library(GenABEL)
library(ParallABEL)
data(ge03d2.clean)
data <- ge03d2.clean[, ]

npro=2 # npro = number of processors
fun=hom

output.p = type2.p(npro,fun,data)

output.p[1:5, ]

mpi.quit(save="no")

#####

Example 2 (summit job on Sun Grid Engine)
You have to create 2 files if you want to submit a job on Sun Grid Engine.
(http://math.acadiau.ca/ACMMaC/Rmpi/submitting.html)

File 1 (R_script.sh):
#####
#!/bin/bash

# Run in the current directory
#$ -cwd

#$ -j y

#$ -V

# Run using bash
#$ -S /bin/bash

# The number of processors required - 1 for frontend-node, plus whatever
# number for compute-nodes.
# This example runs with a processor on frontend-node, plus 2 processors on compute-nodes
#$ -pe lam 3

# Run the job. Replace with whatever R script should run
lamrun -np 1 R --slave CMD BATCH R_script.R R_script_sh.Rout
#####

File 2 (R_script.R):
#####
#clear working space
rm(list = ls())
library(GenABEL)
library(ParallABEL)

```

```

data(ge03d2.clean)
data <- ge03d2.clean[, ]

npro=2 # npro = number of processors
fun=hom

output.p = type2.p(npro,fun,data)

output_Rdata = "output_type2_hom_c3.Rdata"
save(output.p,file=output_Rdata)
# You can load the output from output_type2_hom_c3.Rdata

output.p[1:5, ]

mpi.quit(save="no")

#####

For submit the job please use command:
qsub R_script.sh

You can see the progress on R_script_sh.Rout.

```

---

type3.p

*Parallel for the pairwise statistics derived from analyses between each individual*


---

## Description

Parallel for pairwise statistics derived from analyses between each individual, for example genome-wide identity-by-state or genomic kinship analyses

## Usage

```
type3.p(npro, fun, data, data_f, ...)
```

## Arguments

npro	number of processors on the compute-nodes
fun	function name will be processed such as ibs
data	object of snp.data-class
data_f	file name that was saved the input object of snp.data-class, the object must be named "data"
...	further arguments passed to function of fun argument

## Author(s)

Unitsa Sangket, Yurii S. Aulchenko and Surakameth Mahasirimongkol

**Examples**

```

Example 1 (summit job on R)
#####
#clear working space
rm(list = ls())

library(GenABEL)
library(ParallABEL)
data(srdta)
data <- srdta[,]
npro=2 # npro = number of processors on the compute-nodes
fun=ibs

output.p = type3.p(npro,fun,data)

output.p[1:5,1:5]

mpi.quit(save="no")

#####

Example 2 (summit job on Sun Grid Engine)
You have to create 2 files if you want to submit a job on Sun Grid Engine.
(http://math.acadiau.ca/ACMMaC/Rmpi/submitting.html)

File 1 (R_script.sh):
#####
#!/bin/bash

# Run in the current directory
#$ -cwd

#$ -j y

#$ -V

# Run using bash
#$ -S /bin/bash

# The number of processors required - 1 for frontend-node, plus whatever
# number for compute-nodes.
# This example runs with a processor on frontend-node, plus 2 processors on compute-nodes
#$ -pe lam 3

# Run the job. Replace with whatever R script should run
lamrun -np 1 R --slave CMD BATCH R_script.R R_script_sh.Rout
#####

File 2 (R_script.R):
#####
#clear working space
rm(list = ls())

library(GenABEL)
library(ParallABEL)

```

```

data(srdata)
data <- srdata[,]
npro=2 # npro = number of processors on the compute-nodes
fun=ibs

output.p = type3.p(npro,fun,data)

output_Rdata ="output_type3_ibs_c3.Rdata"
save(output.p,file=output_Rdata)
# You can load the output from output_type3_ibs_c3.Rdata

output.p[1:5,1:5]

mpi.quit(save="no")

#####

For submit the job please use command:
qsub R_script.sh

You can see the progress on R_script_sh.Rout.

```

---

type4.p

*Parallel for the pairwise statistics derived from each pair of SNP*


---

### Description

Parallel for pairwise statistics derived from each pair of SNP, e.g. linkage disequilibrium characterisation

### Usage

```
type4.p(npro, fun, data, data_f, output_f, ...)
```

### Arguments

npro	number of processors on compute nodes
fun	function name will be processed such as r2fast
data	object of snp.data-class
data_f	file name that was saved the input object of snp.data-class, the object must be named "data"
output_f	a file will be saved the outputs
...	further arguments passed to function of fun argument

### Author(s)

Unitsa Sangket, Yurii S. Aulchenko and Surakameth Mahasirimongkol

**Examples**

```

Example 1 (summit job on R)
#####
#clear working space
rm(list = ls())

library(GenABEL)
library(ParallABEL)
data(srdta)
data <- srdta[,]
npro=2 # npro = number of processors
output_f="output_type4_r2fast_c3"
fun=r2fast

output.p = type4.p(npro,fun,data,output_f=output_f)

# You can load the output from output_type4_r2fast_c3.Rdata
mpi.quit(save="no")

#####

Example 2 (summit job on Sun Grid Engine)
You have to create 2 files if you want to submit a job on Sun Grid Engine.
(http://math.acadiau.ca/ACMMaC/Rmpi/submitting.html)

File 1 (R_script.sh):
#####
#!/bin/bash

# Run in the current directory
#$ -cwd

#$ -j y

#$ -V

# Run using bash
#$ -S /bin/bash

# The number of processors required - 1 for frontend-node, plus whatever
# number for compute-nodes.
# This example runs with a processor on frontend-node, plus 2 processors on compute-nodes
#$ -pe lam 3

# Run the job. Replace with whatever R script should run
lamrun -np 1 R --slave CMD BATCH R_script.R R_script_sh.Rout
#####

File 2: R_script.R:
#####
#clear working space
rm(list = ls())

library(GenABEL)
library(ParallABEL)

```



```
data(srdta)
data <- srdta[,]
npro=2 # npro = number of processors
output_f="output_type4_r2fast_c3"
fun=r2fast

output.p = type4.p(npro,fun,data,output_f=output_f)

# You can load the output from output_type4_r2fast_c3.Rdata
mpi.quit(save="no")

#####

For submit the job please use command:
qsub R_script.sh

You can see the progress on R_script_sh.Rout.
```

# Index

type1.p, 1  
type2.p, 3  
type3.p, 5  
type4.p, 7